spm - Secure Package Manager

Srini, rs@toprllc.com

July 22, 2021

1 Introduction

Package managers like **dpkg**, **apt** are ubiquitous and versatile. This projectlet will build a package manager similar to these with security features added. Being somewhat lighter, **spm** is designed to target both applications and for data.

1.1 Use Cases

There are 2 use cases for this tool. In both cases, the need to authenticate and tamper protect the artifacts is the essential feature.

Software Installation Devices that are distributed - for example resident in hospitals could leverage this to perform software upgrades. While the cloud connectivity is the primary enabler, the challenge of authentication of the software upgrades is significant. This tool can play a central role in implementing a secure software upgrade architecture.

Data Archival The devices have the need to archive data - be it configurational or operational data. The security of the data is a persistent concern. Typically devices are configured with customer specific details such as network names, local WiFi credentials and so on which have to be protected from leaking.

Similarly medical devices recording patient diagnostic or therapy data often need to archive for later analysis.

1.2 Learning Objectives

Application configuration with yaml Application configurations increasingly leverage **yaml** (`https://yaml.org/`) in a rather rich field incorporating **xml**, **json** and so on. In this project, the configuration is in the yaml format.

Container Files - tar.gz The goal being the delivery of a set of files to the target, the application supports collecting a set of files, directory structures and so on. Most modern language environments provide appropriate libraries to manipulate a variety of standard formats. In this project, a container file in the **tar.gz** format ie a compressed **tar** file carries the application files.

Tamper resistant design Leveraging ideas of public key cryptography, tamper protecting the contents. Digital signatures of each file is generated and verified before utilizing the contents.

Securing the container file The final container including the payload files and their signatures is in turn encrypted using a password.

2 Specifications

```
$ ./spm -h
Secure package manager helps prepare and distribute packages of applications
and/or data.

Usage:
  spm [command]

Available Commands:
  build       Build a secure package
  help        Help about any command
  install     Install the package
  version     Report version

Flags:
      --config string   config file (default is $HOME/.spm.yaml)
  -h, --help            help for spm
      --keep            keep workarea

Use "spm [command] --help" for more information about a command.
```

Creating a package

```
$ ./spm build --help
Create a secure package based on the configuration file provided.
Optionally push the artifact(s) to a distribution server.
The first argument is the package spec file (ex spec.yaml)
Output package name is the second argument

Usage:
  spm build [flags]

Flags:
  -h, --help   help for build

Global Flags:
      --config string   config file (default is $HOME/.spm.yaml)
      --keep            keep workarea
```

```
 of the artifacts. Argument
        is the package (.spm)
```

Installing a package

```
Usage:
  spm install [flags]

Flags:
  -h, --help   help for install
      --show   extract and show the contents. do not install. Implies --keep

Global Flags:
      --config string   config file (default is $HOME/.spm.yaml)
      --keep            keep workarea
```

2.1 Examples

Sample configuration In the following configuration file, we are asking **spm** to pack files asac.pdf, usb_ddk.pdf, build.sh etc specifying a target installation location for each. In this simplistic example, the **preinstall** and **postinstall** dont ask for anything exciting.

```
$ cat sp.yaml

package:
    name: ServicePack
    version: 1.3

contents:
    - from: acsac.pdf
      to: /tmp/acsac.pdf
    - from: usb_ddk.pdf
      to: /tmp/usbddk.pdf
    - from: build.sh
      to: /tmp/build.sh
    - from: install.sh
      to: /tmp/install.sh

preinstall:
    - go version
    - ls -l /tmp

postinstall:
    - ls /tmp
    - chmod a+x /tmp/install.sh
    - /tmp/install.sh
```

Create a secure package

```
$ cat build.sh
#!/bin/bash
export SPM_PKGPASSWORD=pkgmgr
spm build --config ./spmconfig.yaml --keep ./sp.yaml ./sp.spm

$ ./build.sh
Using config file: ./spmconfig.yaml
Pkg publish url=https://drive.google.com/ Artifacts=https://drive.aws.com/
2021/07/20 06:50:43 Building package for configuration file ./sp.yaml
2021/07/20 06:50:43 Workarea created /tmp/spm677357369
2021/07/20 06:50:43 Created dir /tmp/spm677357369/tmp/spm677357369/contents and /tmp/spm677357369/artifacts
Loaded package File: ./sp.yaml Name : ServicePack
2021/07/20 06:50:43 Copying file acsac.pdf to /tmp/spm677357369/contents
2021/07/20 06:50:43 Copying file usb_ddk.pdf to /tmp/spm677357369/contents
2021/07/20 06:50:43 Copying file build.sh to /tmp/spm677357369/contents
2021/07/20 06:50:43 Copying file install.sh to /tmp/spm677357369/contents
```

Now, a keypair is generated to digitally sign the contents of each file.

```
2021/07/20 06:50:43 Created keypair /tmp/spm677357369/work/private.pem and /tmp/spm677357369/contents/public.pem
2021/07/20 06:50:43 Content file /tmp/spm677357369/contents/acsac.pdf
2021/07/20 06:50:43 Content file /tmp/spm677357369/contents/build.sh
2021/07/20 06:50:43 Content file /tmp/spm677357369/contents/install.sh
2021/07/20 06:50:43 Content file /tmp/spm677357369/contents/public.pem
2021/07/20 06:50:43 Content file /tmp/spm677357369/contents/usb_ddk.pdf
2021/07/20 06:50:43 Files: [/tmp/spm677357369/contents/acsac.pdf /tmp/spm677357369/contents/build.sh /tmp/spm677357369/conte
```

Now signing each of the contents files with the keypair that was generated.

```
2021/07/20 06:50:43 Signing using /tmp/spm677357369/work/private.pem of 5 files
2021/07/20 06:50:43 Signing /tmp/spm677357369/contents/acsac.pdf creating /tmp/spm677357369/contents/acsac.pdf.sig
2021/07/20 06:50:43 Signing /tmp/spm677357369/contents/build.sh creating /tmp/spm677357369/contents/build.sh.sig
2021/07/20 06:50:43 Signing /tmp/spm677357369/contents/install.sh creating /tmp/spm677357369/contents/install.sh.sig
2021/07/20 06:50:43 Signing /tmp/spm677357369/contents/public.pem creating /tmp/spm677357369/contents/public.pem.sig
2021/07/20 06:50:43 Signing /tmp/spm677357369/contents/usb_ddk.pdf creating /tmp/spm677357369/contents/usb_ddk.pdf.sig
2021/07/20 06:50:43 Unique Id created 3d436965-8e29-48a2-96ab-65e3b8f2d12b
2021/07/20 06:50:43 Saved manifest /tmp/spm677357369/contents/Packagefile
2021/07/20 06:50:43 Signing /tmp/spm677357369/contents/Packagefile creating /tmp/spm677357369/contents/Packagefile.sig
2021/07/20 06:50:43 Signed the Package file. Generated /tmp/spm677357369/contents/Packagefile.sig
```

Finally pack the contents into a container file including a manifest **Packagefile**

```
2021/07/20 06:50:43 Created /tmp/spm677357369/work/sp.spm
2021/07/20 06:50:43 Adding Packagefile Size 505
2021/07/20 06:50:43 Adding Packagefile.sig Size 256
2021/07/20 06:50:43 Adding acsac.pdf Size 123519
2021/07/20 06:50:43 Adding acsac.pdf.sig Size 256
2021/07/20 06:50:43 Adding build.sh Size 104
2021/07/20 06:50:43 Adding build.sh.sig Size 256
2021/07/20 06:50:43 Adding install.sh Size 40
2021/07/20 06:50:43 Adding install.sh.sig Size 256
2021/07/20 06:50:43 Adding public.pem Size 418
2021/07/20 06:50:43 Adding public.pem.sig Size 256
2021/07/20 06:50:43 Adding usb_ddk.pdf Size 249204https://www.overleaf.com/project/60f6a1439bd9af3aec74eeb1
2021/07/20 06:50:43 Adding usb_ddk.pdf.sig Size 256
```

Finally, the container file is encrypted using the passphrase. The **passphrase** is read from the environment variable as indicated in the build shell script.

```
2021/07/20 06:50:43 Created /tmp/spm677357369/work/sp.spm
2021/07/20 06:50:43 Encrypt from: /tmp/spm677357369/work/sp.spm to ./sp.spm passphrase pkgmgr
2021/07/20 06:50:43 Created ./sp.spm
```

3 Implementation

3.1 Design considerations

Securing the file The container file is encrypted using the supplied password. The safe-keeping of the password is assumed. Without the password of course the file cannot be decrypted.

File content authentication As shown in the log files, every file that gets packed in the container is accompanied by a cryptographic signature file. The signature itself is generated using the private key of a key pair. The file contents are authenticated by generating another signature with the public part of the keypair which is also included in the container file. If either of the file or the signature file is tampered with, then the whole package will be rejected while installing.

Implementation in **go** `https://github.com/RajaSrinivasan/spm.git`